Lab1

1. You are given n numbers of coin. Supply of each coin is infinite. You are also given an amount m. Your task is to calculate minimum number of coins that are required to make the change of the given amount m.

2. You are given n number of product as input. each product has a particular weight W and value V. Your are also given a sack capacity C. Your task is to find the maximum profit that can be made using the capacity C from the product n.

Lab2

Job Sequencing Problem

Given an array of jobs where every job has a deadline and associated profit if the job is finished before the deadline. It is also given that every job takes the single unit of time, so the minimum possible deadline for any job is 1. How to maximize total profit if only one job can be scheduled at a time.

Input: Four Jobs with following

deadlines and profits

JobID Deadline Profit

a 4 20

b 1 10

c 1 40

d 1 30

Output: Following is maximum

profit sequence of jobs

c, a

Input: Five Jobs with following

deadlines and profits

JobID Deadline Profit

a 2 100

b 1 19

c 2 27

d 1 25

e 3 15

Output: Following is maximum

profit sequence of jobs

c, a, e

A Simple Solution is to generate all subsets of given set of jobs and check individual subset for feasibility of jobs in that subset. Keep track of maximum profit among all feasible subsets. The time complexity of this solution is exponential. This is a standard Greedy Algorithm problem. Following is the algorithm.

1) Sort all jobs in decreasing order of profit.

2) Iterate on jobs in decreasing order of profit. For each job , do the following :

a) Find a time slot i, such that slot is empty and i < deadline and i is greatest. Put the job in This slot and mark this slot filled.

b) If no such i exists, then ignore the job.

2. Job Sequencing Problem – Loss Minimization

We are given N jobs numbered 1 to N. For each activity, let Ti denotes the number of days required to complete the job. For each day of delay before starting to work for job i, a loss of Li is incurred.

We are required to find a sequence to complete the jobs so that overall loss is minimized. We can only work on one job at a time.

Input : L = {3, 1, 2, 4} and

T = {4, 1000, 2, 5}

Output : 3, 4, 1, 2

Explanation: We should first complete

Job 3, then jobs 4, 1, 2 respectively.

Input : L = {1, 2, 3, 5, 6}

T = {2, 4, 1, 3, 2}

Output : 3, 5, 4, 1, 2

Explanation: We should complete jobs

3, 5, 4, 1 and then 2 in this order.

Let us consider two extreme cases and we shall deduce the general case solution from them.

1. All jobs take same time to finish, i.e. Ti = k for all i. Since all jobs take same time to finish we should first select jobs which have large Loss (Li). We should select jobs which have the highest losses and finish them as early as possible. Thus this is a greedy algorithm. Sort the jobs in descending order based on Li only.

2. All jobs have the same penalty. Since all jobs have the same penalty we will do those jobs first which will take less amount of time to finish. This will minimize the total delay, and hence also the total loss incurred. This is also a greedy algorithm. Sort the jobs in ascending order based on Ti. Or we can also sort in descending order of 1/Ti.

From the above cases, we can easily see that we should sort the jobs not on the basis of Li or Ti alone. Instead, we should sort the jobs according to the ratio Li/Ti, in descending order.

3. Minimum Number of Platforms Required for a Railway

Given arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays which represent arrival and departure times of trains that stop.

Input: arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}

dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}

Output: 3

Explanation: There are at-most three trains at a time (time between 11:00 to 11:20)

Input: arr[] = {9:00, 9:40}

dep[] = {9:10, 12:00}

Output: 1

Explanation: Only one platform is needed.

Lab3

1. Activity selection problem: Given a set of activities, along with the starting and finishing time of

each activity, find the maximum number of activities performed by a single person assuming that a person can only work on a single activity at a time.

Example

3,7

2,4

5,8

6,9

1,11

10,12

0,3

Output: 3

2. Minimize the sum of Product: You are given two arrays, A and B, of equal size N. The task is to find the minimum value of A[0] * B[0] + A[1] * B[1] +...+ A[N-1] * B[N-1], where shuffling of elements of arrays A and B is allowed.

Example 1:

Input:

N = 3

A[] = {3, 1, 1}

B[] = {6, 5, 4}

Output:

23

Explanation:

1*6+1*5+3*4 = 6+5+12

= 23 is the minimum sum

Example 2:

Input:

N = 5

A[] = {6, 1, 9, 5, 4}

B[] = {3, 4, 8, 2, 4}

Output:

80

Explanation:

2*9+3*6+4*5+4*4+8*1

=18+18+20+16+8

= 80 is the minimum sum

3. Assigning mice to hole: There are N Mice and N holes are placed in a straight line. Each hole can accommodate only 1 mouse. A mouse can stay at his position, move one step right from x to x + 1, or move one step left from x to x -1. Any of these moves consumes 1 minute. Assign mice to holes so that the time when the last mouse gets inside a hole is minimized.

Example:

Input: positions of mice are:

4 -4 2

Positions of holes are:

4 0 5

Output: 4

Assign mouse at position x = 4 to hole at

Position x = 4 : Time taken is 0 minutes

Assign mouse at position x=-4 to hole at

Position x = 0 : Time taken is 4 minutes

Assign mouse at position x=2 to hole at

Position x = 5 : Time taken is 3 minutes

After 4 minutes all of the mice are in the holes.

Since, there is no combination possible where

the last mouse's time is less than 4,

Answer = 4.

Input: positions of mice are:

-10, -79, -79, 67, 93, -85, -28, -94

Positions of holes are:

-2, 9, 69, 25, -31, 23, 50, 78

Output: 102

Lab4

Flow:

- LCS: Longest Common Subsequence (Problem Statement)
- Recursive solution
- Recursive DP (bottom up)
- Iterative DP (top down)
- Trace back
- Longest repeating subsequence

Lab5

1. **0-1 Knapsack**: you are given the weights and value of n item. You are also given the capacity W of a knapsack. You have to find the maximum profits that can be made using the given knapsack size. You can't take product partially. Also show the product list you have taken. Design both iterative and recursive dp for it.

   Input:

   value = [20, 5, 10, 40, 15, 25]

   weight = [1, 2, 3, 8, 7, 4]

   int W = 10

Lab6

1. Take an undirected and unweighted graph using adjacency matrix. Apply DFS to determine whether the graph is connected or not.
   2. Take an undirected and unweighted graph using adjacency matrix as input. Apply BFS algorithm to find path from source node to all other node.

Input
4 5
0 1
0 3
1 2
2 3
1 3
source: 0
Output
0->1
0->1->2
0->3

Lab7

1. Take a weighted graph as input. Find shortest cost from source node to all other nodes using Bellman ford algorithm.
   2. Find shortest path from source to all other node.
   for example
   shortest path from node 0 to node 1 be like
   0->3->2->1

Lab8

Take an undirected and weighted graph as input. Apply Kruskal Algorithm to find minimum spanning tree from the graph.

Lab9

- Implement Floyd Warshall algorithm in a weighted directed graph to find all pair shortest paths.
- Print the all pair shortest paths.

Makeup lab on DP

1. The Shortest Common Supersequence (SCS) is finding the shortest super sequence Z of given

sequences X and Y such that both X and Y are subsequences of Z.

For example, consider the two following sequences, X and Y:

X: ABCBDAB

Y: BDCABA

The length of the SCS is 9

The SCS are ABCBDCABA, ABDCABDAB, and ABDCBDABA

2. Given an unlimited supply of coins of given denominations, find the minimum number of coins required to get the desired change.

For example, consider S = { 1, 3, 5, 7 }.

If the desired change is 15, the minimum number of coins required is 3

(7 + 7 + 1) or (5 + 5 + 5) or (3 + 5 + 7)

If the desired change is 18, the minimum number of coins required is 4

(7 + 7 + 3 + 1) or (5 + 5 + 5 + 3) or (7 + 5 + 5 + 1)